AML Summary

Lukas Diebold

December 15, 2025

Contents

1 Math Preliminaries

Gibbs Distribution The Gibbs distribution is a probability distribution that assigns likelihoods to states based on a cost function, with lower-cost states being more probable. Given a set of states $x \in \mathcal{X}$, a cost function $E(x)$, and an inverse temperature parameter $\beta > 0$, the Gibbs distribution is

$$p(x) = \frac{1}{Z} e^{-\beta E(x)}$$

where the partition function Z ensures normalization

$$Z = \sum_{x \in X} e^{-\beta E(x)}$$

2 Conceptual Foundation

2.1 What is Machine Learning?

"ML is a mathematization of epistemolagy!". In philosophy, it is the science of knowledge, the science of what can be known. This is relevant, because in ML we are interested in systems that produce/generate knowledge.

The goal is then to observe 'reality' and draw conclusions from the observations. This can be seen as a perception-action cycle. Where perception is the result of our observations (typically in a data space $\mathcal{X}$) and the actions are part of a hypothesis space. To go from the data space to the hypothesis class $\mathcal{C}$ we generally use an algorithm A. See Figure 1 for an overview.

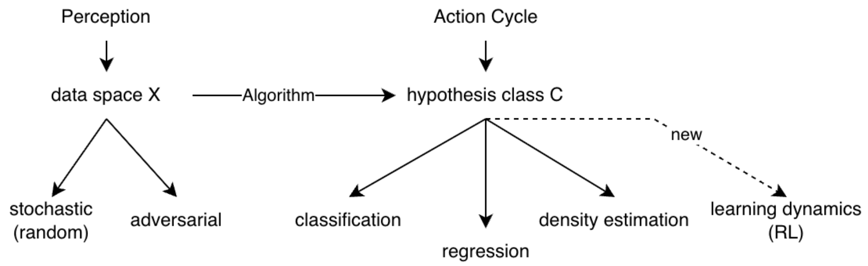

Figure 1: Overview ML

Information processing occurs when $|\mathcal{X}| \gg |\mathcal{C}|$. Taking the example of combinatorial optimization problems, $|\mathcal{X}|$ would be the space of weighted graphs, and we look for a color of the graph or something similar. Then $|\mathcal{X}| \approx K^{\binom{n}{2}}$ and $|\mathcal{C}| \approx e^{n \log n}$ so we obserge that $|\mathcal{X}|$ is much larger.

2.2 Conceptual foundation of inference

1. Perception of realits is mediated by data of senses/sensors
2. Data are stochastic $\rightarrow$ porbabilistic
3. Sensing restricts us to selected aspects of reality
4. Humans interpret data by a huge reduction in degrees of freedom
   $|x| \gg |e|$ (space of graphs $\gg$ space of colorings, cycles, spanning trees)
5. Tuple ({data }, {hypotheses }) define models:

$$\begin{aligned} \mathcal{A} : \mathcal{X} &\longrightarrow \mathcal{C} \\ x &\longmapsto c = \mathcal{A}(x) \end{aligned}$$

6. Experiments ϵ provide us with data
7. Learning means interpreting X w.r.t hypotheses $\mathcal{C}$

2.3 Artificial Intelligence

Taking a high level view. We live in very high dimensional data, which we are not able to fully process. We use algorithms to make sense of the data, and this then informs our values, from this we get the following relation

$$\text{Data} \longrightarrow \text{ Algorithms } \mathcal{A} \longrightarrow \text{ Values}$$

In epistemology, we differentiate between **deduction** and **induction**. Deduction is a form of reasoning in which the conclusion follows necessarily from the premises, while induction tries to generalize, that is, the conclusion goes beyond the premises and generally probabilistic. We can construct a model in which deduction and induction form a **feedback loop**, not two isolated methods. In simpler terms, on one side we try to formulate axioms from our observations (empirical data), and on the other side we then use these axioms and derive logical consequences from them. This is not anything new, and this cycle generally informs the model of "Theory, Experiment, Computation" in science. What has changed with ML/AI is that we're now in the era of non-parametric modeling.

2.4 Extracting Value from Data - What is the problem?

- Algorithms that process inputs with noise compute random variables as outputs!
- Algorithms should compute typical solutions!
- When do algorithms generalize over noise/model mismatch?
- How can algorithms autonomously improve performance?

2.5 What does Generalization mean?

- Out-of-sample risk

  $$\theta^\star(X') \sim \mathbb{P}^{\mathcal{A}}(\theta \mid X) \in \arg\min_{\mathbb{P}(.|.)} \mathbb{E}_{X'}\mathbb{E}_{\theta|X'}\mathbb{E}_{X''|X'} R(\theta, X'')$$

  where X' is the training data and X'' is the test data, so thiw is the risk where θ is conditioned on the training data (trained the model).

- Log loss of posterior (risks and probabilities are dependent!)

  $$\begin{aligned}\theta^\star(X') \sim \mathbb{P}^{\mathcal{A}}(\theta \mid X) &\in \arg\min_{\mathbb{P}(.|.)} \mathbb{E}_{X'}\mathbb{E}_{\theta|X'}\mathbb{E}_{X''|X'} \left(-\log \mathbb{P}(\theta \mid X'')\right) \\ &\in \arg\min_{\mathbb{P}(.|.)} \mathbb{E}_{X'}\mathbb{E}_{\theta|X'}\mathbb{E}_{X''|X'} \left(\beta R(\theta, X'') + \log Z\right)\end{aligned}$$

- Posterior agreement

  $$\theta^\star(X') \sim \mathbb{P}^{\mathcal{A}}(\theta \mid X) \in \arg\min_{\mathbb{P}(.|.)} \mathbb{E}_{X'}\mathbb{E}_{X''|X'} \big(-\log \mathbb{E}_{\theta|X'}\mathbb{P}(\theta \mid X'')\big)$$

2.6 Conceptional Foundation of Inference

- Our perception of “our world” / reality is mediated by data of senses / sensors.
- Our data are influenced by chance.
- Creatures interpret selected aspects of reality by hypotheses to "survive and reproduce"
- Data and hypotheses define models to enable judgements, decisions and actions.
- AI / ML: Algorithms define relations of data and hypotheses, e.g., they select models !

3 Fundamentals of Machine Learning

Machine learning is fundamentally about inferring models from data. At its core lies Bayes' rule, which relates the posterior distribution (model given data) to the likelihood and prior:

$$\mathbb{P}(\text{model} \mid \text{data}) = \frac{\mathbb{P}(\text{ data } \mid \text{ model })\mathbb{P}(\text{ model })}{\mathbb{P}(\text{ data })}$$

In the **maximum likelihood (ML)** approach, we select the model that maximizes the likelihood of observing the data:

$$\widehat{\text{model}} \in \arg\max_{\text{model}} \mathbb{P}(\text{data} \mid \text{model})$$

Under regularity conditions, the ML estimator $\widehat{\text{model}_n}$ is consistent, asymptotically normal, and asymptotically efficient.

Consistency: A point estimator $\hat{\theta}_n$ of the parameter $\theta = \theta_0$ is consistent if it converges in probability to the true parameter:

$$\forall \varepsilon > 0, \mathbb{P}\left(\left|\hat{\theta}_n - \theta_0\right| > \varepsilon\right) \xrightarrow{n\to\infty} 0$$

More formally, using the ε-δ definition:

$$\forall \theta, \forall \varepsilon, \delta > 0, \exists n_0, \forall n > n_0, \mathbb{P}\left(\left|\widehat{\theta}_n - \theta\right| < \varepsilon\right) > 1 - \delta$$

Efficiency: An estimator $\hat{\theta}_n$ is efficient if it achieves the minimum mean squared error among all estimators:

$$\hat{\theta}_n = \arg\min_{\hat{\theta}} \mathbb{E}\left[\left(\hat{\theta}_n - \theta_0\right)^2\right]$$

The question then arises: how precisely can we estimate θ given n samples? The Cramér-Rao bound provides a fundamental lower bound on the variance of any unbiased estimator.

3.1 Efficiency: Cramér-Rao Bound

Problem: What is the best achievable precision for parameter estimation?

Given a likelihood $p(y \mid \theta)$ for $\theta \in \Theta$ and data $y_1, \ldots, y_n \sim p(y \mid \theta = \theta_0)$, we ask: How precisely can we estimate $\theta = \theta_0$ given n samples?

For an estimator $\hat{\theta}(y_1, \ldots, y_n)$, we measure precision via the mean squared error:

$$\mathbb{E}_{y|\theta}\left[(\hat{\theta} - \theta)^2\right]$$

Key definitions:

- Score: $\Lambda = \frac{\partial}{\partial \theta} \log p(y \mid \theta) = \frac{\frac{\partial}{\partial \theta} p(y|\theta)}{p(y|\theta)}$
- Bias: $b_{\hat{\theta}} = \mathbb{E}_{y|\theta}\left[\hat{\theta}(y_1, \ldots, y_n)\right] - \theta$

Expected score: The score has zero mean.

$$\begin{aligned}\mathbb{E}_{y|\theta}[\Lambda] &= \int p(y \mid \theta) \frac{\frac{\partial}{\partial \theta} p(y \mid \theta)}{p(y \mid \theta)} dy \\ &= \frac{\partial}{\partial \theta} \int p(y \mid \theta) dy = \frac{\partial}{\partial \theta} 1 = 0\end{aligned}$$

Score-estimator product:

$$\begin{aligned}\mathbb{E}_{y|\theta}[\Lambda\hat{\theta}] &= \int p(y \mid \theta) \frac{\frac{\partial}{\partial \theta} p(y \mid \theta)}{p(y \mid \theta)} \hat{\theta} dy \\ &= \frac{\partial}{\partial \theta}\left(\int p(y \mid \theta)\hat{\theta} dy\right) \\ &= \frac{\partial}{\partial \theta}\left(\mathbb{E}_{y|\theta}\hat{\theta}\right) = \frac{\partial}{\partial \theta}\left(b_{\hat{\theta}} + \theta\right) = \frac{\partial}{\partial \theta} b_{\hat{\theta}} + 1\end{aligned}$$

Cross-correlation:

$$\mathbb{E}_{y|\theta}\left[(\Lambda-\mathbb{E}\Lambda)\left(\hat{\theta}-\mathbb{E}\hat{\theta}\right)\right]=\mathbb{E}_{y|\theta}[\Lambda\hat{\theta}]-\mathbb{E}_{y|\theta}[\Lambda]\mathbb{E}\hat{\theta}=\mathbb{E}_{y|\theta}[\Lambda\hat{\theta}]$$

since $\mathbb{E}[\Lambda]=0$.

Cauchy-Schwarz inequality: Applying Cauchy-Schwarz to the cross-correlation:

$$\left(\mathbb{E}_{y|\theta}[\Lambda(\hat{\theta}-\mathbb{E}\hat{\theta})]\right)^2\leq\mathbb{E}_{y|\theta}\left[\Lambda^2\right]\mathbb{E}_{y|\theta}\left[(\hat{\theta}-\mathbb{E}\hat{\theta})^2\right]$$

Expanding the right-hand side:

$$\begin{aligned}\mathbb{E}_{y|\theta}\left[(\hat{\theta}-\mathbb{E}\hat{\theta})^2\right]&=\mathbb{E}_{y|\theta}\left[(\hat{\theta}-\theta+\theta-\mathbb{E}\hat{\theta})^2\right]\\&=\mathbb{E}_{y|\theta}\left[(\hat{\theta}-\theta)^2\right]-b_{\hat{\theta}}^2\end{aligned}$$

Therefore:

$$\left(\frac{\partial}{\partial\theta}b_{\hat{\theta}}+1\right)^2\leq\mathbb{E}_{y|\theta}\left[\Lambda^2\right]\left(\mathbb{E}_{y|\theta}\left[(\hat{\theta}-\theta)^2\right]-b_{\hat{\theta}}^2\right)$$

Rearranging yields the **Cramér-Rao bound**:

$$\mathbb{E}_{y|\theta}\left[(\hat{\theta}-\theta)^2\right]\geq\frac{\left(\frac{\partial}{\partial\theta}b_{\hat{\theta}}+1\right)^2}{\mathbb{E}_{y|\theta}\left[\Lambda^2\right]}+b_{\hat{\theta}}^2$$

Fisher information: The expected squared score is called the Fisher information:

$$I(\theta):=\mathbb{E}_{y|\theta}\left[\Lambda^2\right]=\int p(y\mid\theta)\left(\frac{\partial}{\partial\theta}\log p(y\mid\theta)\right)^2dy$$

It measures how much information the data contains about the parameter θ. Higher Fisher information means we can estimate θ more precisely.

Remarks:

- For unbiased estimators ($b_{\hat{\theta}}=0$), the bound simplifies to $\mathbb{E}[(\hat{\theta}-\theta)^2]\geq 1/I(\theta)$.
- The bound reveals a trade-off for biased estimators: reducing bias derivative $\frac{\partial}{\partial\theta}b_{\hat{\theta}}$ vs. reducing squared bias $b_{\hat{\theta}}^2$. Unbiased estimators are not always optimal!

Fisher information for n i.i.d. samples:

$$\begin{aligned}I^{(n)}(\theta)&=\mathbb{E}_{y_1,\ldots,y_n|\theta}\left[\Lambda^2\right]\\&=\mathbb{E}\left[\left(\frac{\partial}{\partial\theta}\log p\left(y_1,\ldots,y_n\mid\theta\right)\right)^2\right]\\&=\mathbb{E}\left[\left(\sum_{i=1}^n\frac{\partial}{\partial\theta}\log p\left(y_i\mid\theta\right)\right)^2\right]=\mathbb{E}\left[\left(\sum_{i=1}^n\Lambda_i\right)^2\right]\\&=\sum_{i=1}^n\mathbb{E}\left[\Lambda_i^2\right]+\sum_{i\neq j}\mathbb{E}\left[\Lambda_i\right]\mathbb{E}\left[\Lambda_j\right]\\&=\sum_{i=1}^n\mathbb{E}\left[\Lambda_i^2\right]=nI(\theta)\end{aligned}$$

where the cross-terms vanish because $\mathbb{E}[\Lambda_i]=0$ and the samples are independent.

Key insight: The Fisher information of n i.i.d. random variables is n times the Fisher information of a single random variable. This shows that precision improves linearly with sample size.

4 Regression

4.1 Act 1: High-dimensional regression is unstable

We assume X and y are distributed according to a distribution p_* (i.e. $X, y \sim p_*$), where the output follows a noisy linear model:

$$y = f_*(x) + \varepsilon \quad \text{with } \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

Here f_* is the true (unknown) regression function and ε is additive Gaussian noise with variance σ^2. Our task is to estimate f_* from training data $D = \{(x_i, y_i)\}_{i=1}^n \sim p_*$.

The problem in this form is not tractable because the space of all possible functions is too large. We therefore restrict ourselves to linear functions:

$$f_*(x) = \beta^\top x$$

where $\beta \in \mathbb{R}^d$ is a parameter vector. Given the Gaussian noise assumption, each observation has likelihood $p(y_i|x_i, \beta) = \mathcal{N}(\beta^\top x_i, \sigma^2)$. We solve for β using Maximum Likelihood Estimation (MLE):

$$\begin{aligned}
\widehat{\beta} &= \arg\max_{\beta \in \mathbb{R}^d} p(D \mid \beta) \\
&= \arg\max_{\beta} \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \beta^\top x_i)^2}{2\sigma^2}\right) \\
&= \arg\min_{\beta} \sum_{i=1}^n (y_i - \beta^\top x_i)^2 \\
&= \arg\min_{\beta} \mathrm{MSE}(D, \beta)
\end{aligned}$$

Maximizing the log-likelihood is equivalent to minimizing the mean squared error (MSE). The closed-form solution depends on whether we have more features than samples or vice versa:

$$\begin{aligned}
\widehat{\beta} &= (X^\top X)^{-1} X^\top y \quad \text{(when } d < n\text{, more samples than features)} \\
&= X^\top (X X^\top)^{-1} y \quad \text{(when } d > n\text{, more features than samples)}
\end{aligned}$$

These are algebraically equivalent by the Woodbury matrix identity. The first formula is the standard *ordinary least squares (OLS)* estimator, where

$$X = \begin{bmatrix} -x_1- \\ \vdots \\ -x_n- \end{bmatrix} \quad y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

This estimator has some interesting properties. It is unbiased and, by the Gauss-Markov Theorem, it is the best linear unbiased estimator (BLUE), i.e. it attains the smallest variance among all linear unbiased estimators. Thus, from the formula

$$\text{error} = \text{bias}^2 + \text{variance} + \text{noise}$$

we find that this estimator is the one with the smallest error of all the unbiased estimators. Then why does no-one use this estimator? If we introduce a bit of bias, we can significantly reduce the variance.

To understand the instability, we analyze $\mathrm{Var}(\hat{\beta})$ using the singular value decomposition (SVD) $X = UDV^\top$, where $U \in \mathbb{R}^{n \times n}$ and $V \in \mathbb{R}^{d \times d}$ are orthogonal, and D is diagonal with singular values $D_{11} \geq D_{22} \geq \ldots \geq 0$. Plugging this into the OLS formula:

$$\hat{\beta} = (X^\top X)^{-1} X^\top y = (VD^\top U^\top UDV^\top)^{-1} VD^\top U^\top y = VD^{-1}U^\top y$$

Since $y = X\beta_* + \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$, and we multiply y by the deterministic matrix $VD^{-1}U^\top$, the estimator $\hat{\beta}$ is also Gaussian. Its variance is:

$$\mathrm{Var}(\hat{\beta}) = \mathrm{Var}(VD^{-1}U^\top y) = VD^{-1}U^\top \,\mathrm{Var}(y) UD^{-1}V^\top = \sigma^2 VD^{-2}V^\top = \sigma^2 \sum_{i \leq r} \frac{1}{D_{ii}^2} V_i V_i^\top$$

where $r = \text{rank}(X)$ and V_i is the i-th column of V (the i-th right singular vector).

The problem: In high-dimensional data, features are often correlated (e.g., pixel intensities in images, gene expressions). This makes X close to low-rank, so several singular values D_{ii} are very small. The variance contributions $1/D_{ii}^2$ then explode for these directions, causing massive instability in $\hat{\beta}$ even though it remains unbiased. Small noise in y gets amplified enormously in directions with small singular values, leading to wild predictions on test data.

4.2 Act 2: Stability via Regularization

The solution to the variance blow-up is to introduce regularization, which adds a small amount of bias in exchange for a large reduction in variance. We can derive regularization naturally from a Bayesian perspective.

The typical process of **Bayesian inference** goes through the following stages:

1. Prior $\beta \sim \mathcal{N}(0, \tau^2 I)$ — we assume β is drawn from a Gaussian centered at zero with variance τ^2. This encodes our belief that coefficients should not be too large.
2. Likelihood $p(D|\beta) = \prod_i \mathcal{N}(y_i|\beta^\top x_i, \sigma^2)$ — same Gaussian noise model as before.
3. Posterior (via Bayes' rule) $p(\beta|D) \propto p(\beta)p(D|\beta) \propto \exp\left(-\frac{1}{2\sigma^2}\text{MSE}(D,\beta) - \frac{1}{2\tau^2}\|\beta\|^2\right)$

The posterior combines the likelihood (fit to data) with the prior (regularization). Taking the negative log gives the objective:

$$\hat{\beta}_{\text{MAP}} = \arg\min_\beta \left[\frac{1}{2\sigma^2}\sum_i (y_i - \beta^\top x_i)^2 + \frac{1}{2\tau^2}\|\beta\|^2\right]$$

This is precisely **ridge regression** with regularization parameter $\lambda = \sigma^2/\tau^2$. The ℓ^2 penalty $\|\beta\|^2$ shrinks coefficients toward zero. If we instead use a Laplace prior $p(\beta) \propto \exp(-|\beta|/\tau)$ with heavier tails, we obtain **lasso regression** with an ℓ^1 penalty $\|\beta\|_1$, which promotes sparsity.

The prior variance τ^2 controls the bias-variance trade-off: small τ^2 means strong regularization (more bias, less variance), while large τ^2 recovers OLS (no bias, high variance). The MAP (maximum a posteriori) solution is:

$$\hat{\beta}_{\text{MAP}} = \left(X^\top X + \frac{\sigma^2}{\tau^2} I\right)^{-1} X^\top y$$

Compare this to OLS: the regularization term $\frac{\sigma^2}{\tau^2}I$ is added to $X^\top X$ before inversion, preventing ill-conditioning. Using SVD again to analyze the variance:

$$\text{Var}(\hat{\beta}_{\text{MAP}}) = \sigma^2 \sum_{i \le r} \frac{D_{ii}^2}{\left(D_{ii}^2 + \frac{\sigma^2}{\tau^2}\right)^2} V_i V_i^\top$$

The key is the **shrinkage factor** $\frac{D_{ii}^2}{(D_{ii}^2+\sigma^2/\tau^2)^2}$. For large singular values ($D_{ii}^2 \gg \sigma^2/\tau^2$), this is close to $1/D_{ii}^2$ (like OLS). For small singular values ($D_{ii}^2 \ll \sigma^2/\tau^2$), the factor is approximately $\tau^4 D_{ii}^2/\sigma^4$, which decays much more slowly than $1/D_{ii}^2$. This prevents variance blow-up in the problematic low-variance directions, stabilizing the estimator at the cost of introducing bias (shrinking coefficients toward zero).

4.3 Act 3: Polynomial regression via kernels

Now we change our assumption for $f_*(x)$ to allow for nonlinear functions. We model f_* as a linear function in an infinite-dimensional feature space:

$$f_*(x) = \varphi(x)^\top \beta_*$$

where $\beta_* \in \mathbb{R}^\infty$ and $\varphi(x)$ maps each input to an infinite-dimensional polynomial feature representation:

$$\varphi(X) = K_x \left(\frac{x_1^{\alpha_1} \ldots x_d^{\alpha_d}}{\sqrt{\alpha_1! \ldots \alpha_d!}}\right)_{\alpha \in \mathbb{N}^d}$$

This includes all polynomial terms of all degrees. The normalization by factorials ensures the inner product has a clean closed form.

Remarkably, the inner product of two infinite-dimensional feature vectors yields the radial basis function (RBF) kernel. For $x, x' \in \mathbb{R}^a$:

$$\begin{aligned} \varphi(x)^\top \varphi(x') &= K_{RBF}(x, x') \\ &= \exp\left(-\frac{1}{2}||x - x'||^2\right) \end{aligned}$$

This follows from the Taylor expansion of the exponential function. The RBF kernel measures similarity: it is 1 when $x = x'$ and decays as points move apart.

We still want to minimize the MSE, but now in the infinite-dimensional feature space:

$$\begin{aligned} \hat{\beta} &= \arg\min_{\beta \in \mathbb{R}^\infty} \frac{1}{n} \sum_{i \leq n} \left(y_i - \varphi(x_i)^\top \beta\right)^2 \\ &= \Phi^\top \left(\Phi\Phi^\top\right)^{-1} y \end{aligned}$$

where

$$\Phi = \begin{bmatrix} \varphi(x)^\top \\ \varphi(x_2)^\top \\ \varphi(x_n)^\top \end{bmatrix} \in \mathbb{R}^{n \times \infty}$$

Despite β living in infinite dimensions, the representer theorem guarantees the solution lies in the span of the training features, so we can work with the $n \times n$ Gram matrix $\Phi\Phi^\top$ instead of the infinite-dimensional feature space directly.

To make a prediction at test point x_*, we compute:

$$\begin{aligned} \hat{y}_* &= \varphi(x_*)^\top \hat{\beta} \\ &= \varphi(x_*)^\top \Phi^\top \left(\Phi\Phi^\top\right)^{-1} y \\ &= k(x_*)^\top K^{-1} y \end{aligned}$$

where $k(x_*) = \left(\varphi(x_*)^\top \varphi(x_i)\right)_{1 \leq i \leq n} = (K_{RBF}(x_*, x_i))_{1 \leq i \leq n}$ is an n-dimensional vector of kernel evaluations between the test point and each training point, and $K_{ij} = \varphi(x_i)^\top \varphi(x_j) = K_{RBF}(x_i, x_j)$ is the $n \times n$ kernel matrix.

This is the **kernel trick**: we never explicitly construct the infinite-dimensional $\varphi(\cdot)$. Instead, we only compute inner products via the kernel function K_{RBF}, which can be evaluated in closed form. The prediction is a weighted combination of training outputs, where the weights depend on how similar the test point is to each training point.

The problem is that the inversion of the matrix is $O(n^3)$, which becomes costly for large datasets even though we avoided the infinite feature map explicitly.

4.4 Act 4: Neural Networks

We assume f_* has only a single, very wide hidden layer.

$$f_*(X) = \frac{1}{\sqrt{m}} \sum_{i \leq m} \alpha_i \phi\left(\omega_i^\top X\right)$$

where ϕ is a nonlinear activation function (e.g. ReLU, tanh), and the network has m hidden units. The parameters are $\theta = \{\alpha_i, w_i\}_{i \leq m}$, i.e. both the output weights α_i and the input weights w_i are learned. We initialize with

$$\theta_0 \sim \mathcal{N}\left(0, w^2\right)$$

and we update our parameters using gradient descent.

$$\theta_{t+1} \leftarrow \theta_t - \eta \nabla_\theta \, \mathrm{MSE}(D, \theta_t)$$

The gradient can be written in matrix form as

$$\nabla_\theta \, \mathrm{MSE}(D, \theta_t) = \widetilde{\Phi}_t^\top (f_t - y)$$

where

$$\tilde{\Phi}_t = \left(-\nabla_\theta f\left(x_i; \theta_t\right)^\top - \right)_{i \leq n} \in \mathbb{R}^{n \times |\theta|} \quad \text{and} \quad f_t = \left(f(x_i; \theta_t)\right)_{i \leq n} \in \mathbb{R}^n$$

Here $\tilde{\Phi}_t$ is the feature matrix whose i-th row is the gradient of the network output with respect to all parameters, evaluated at data point x_i and current parameters θ_t.

In the *lazy training regime* (small learning rate, wide network), the parameters stay close to initialization, so we can linearize the network via a first-order Taylor expansion around θ_0:

$$f_t \approx f_0 + \tilde{\Phi}_0 \left(\theta_t - \theta_0\right)$$

Assuming the feature matrix $\tilde{\Phi}_t$ remains approximately constant at $\tilde{\Phi}_0$ (which holds when $m \to \infty$), gradient flow yields

$$\theta_t - \theta_0 = \tilde{\Phi}_0^\top \left(\tilde{\Phi}_0 \tilde{\Phi}_0^\top \right)^{-1} \left(f_t - f_0\right)$$

This says the parameter change lies in the span of the gradients and is chosen to optimally fit the training residuals.

Now let x_* be a test point. Plugging the linearization into the prediction yields

$$f_t\left(x_*\right) \approx f_0\left(x_*\right) + \nabla_\theta f\left(x_*, \theta_0\right)^\top \tilde{\Phi}_0^\top \left(\tilde{\Phi}_0 \tilde{\Phi}_0^\top \right)^{-1} \left(f_t - f_0\right)$$

Define the *neural tangent kernel (NTK)* K with entries

$$K_{ij} = \nabla_\theta f(x_i, \theta_0)^\top \nabla_\theta f(x_j, \theta_0) = \left[\tilde{\Phi}_0 \tilde{\Phi}_0^\top \right]_{ij}$$

and similarly $k(x_*) = \left(\nabla_\theta f(x_*, \theta_0)^\top \nabla_\theta f(x_i, \theta_0) \right)_{i \leq n}$.

In the infinite-width limit ($m \to \infty$), the random initialization ensures $f_0(x) \to 0$ for all x (the outputs average out), and after infinite training time ($t \to \infty$), gradient descent drives the training residual to zero so $f_t \to y$. The prediction becomes

$$f_\infty\left(x_*\right) = k\left(x_*\right)^\top K^{-1} y$$

This is exactly the result we obtained in Act 3 for kernel regression.

Conclusion: Gradient descent on a very wide neural network operates in a *kernel regime*, where training is equivalent to kernel ridge regression with the neural tangent kernel. The NTK is determined by the architecture and activation function, but the solution has the same closed-form structure $k(x_*)^\top K^{-1} y$ as any other kernel method. In practice, finite-width networks can escape this regime and learn richer, feature-learning representations—this lazy limit is a useful theoretical baseline.